Side Channel Modeling Attacks on 65nm Arbiter PUFs Exploiting CMOS Device Noise

Jeroen Delvaux and Ingrid Verbauwhede

ESAT/SCD-COSIC and iMinds, KU Leuven
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
Email: {firstname.lastname}@esat.kuleuven.be

***Abstract*—Physically Unclonable Functions (PUFs) are emerging as hardware security primitives. For so-called strong PUFs, the number of challenge-response pairs (CRPs) increases exponentially with the required chip area in the ideal case. They can provide a mechanism to authenticate chips which is inherently unique for every manufactured sample. Modeling of the CRP behavior through Machine Learning (ML) has shown to be a threat however. In this paper, we exploit repeatability imperfections of PUF responses as a side channel for model building. We demonstrate that 65nm CMOS arbiter PUFs can be modeled successfully, without utilizing any ML algorithm. Data originates from real-world measurements and hence not from simulations. Modeling accuracies exceeding 97% are obtained, which is comparable with previously published ML results. Information leakage through the exploited side channel should be considered for all strong PUF designs. Combined attack strategies, whereby repeatability measurements facilitate ML, might be effective and are recommended for further research.**

I. Introduction

There is a clear trend towards small, distributed, mobile and wireless applications. They are typically integrated on chip. Cryptographic protection is indispensable as almost all applications process sensitive data, but is thwarted because of the trend above. Energy/power and chip area are scarce resources, so we are often limited to lightweight cryptography. Furthermore, because of the mobility, one can easily gain physical access to the chip. Hardware attacks, either invasive or noninvasive, are thus a significant threat.

Classical cryptography heavily relies on the ability to store secret information. Typically through binary storage of keys in non-volatile memory, which is vulnerable to hardware attacks. Also because of the permanent nature of storage which does not pose limits on the time frame of the attacker. Circuits that detect hardware invasion offer additional protection. Unfortunately they suffer from practical limitations. They might be expensive, bulky, battery powered, avoidable and/or not appropriate for lightweight environments.

Physically Unclonable Functions (PUFs) have been proposed as a more secure and more efficient alternative. PUFs measure the unique variability of physical objects. They can be manufactured in a variety of technologies: optical, acoustical, magnetical, electrical and so on. PUFs which can be integrated on chip, especially in CMOS technology, are by far the most relevant for commercial applications. The manufacturing variability of nanoscale structures is then measured.

PUFs are functions and produce a response when queried with a challenge. Responses and challenges are both binary vectors at the highest abstraction level. PUFs are often subdivided in two classes, depending on the number of challenge-response pairs (CRPs) [9]. Weak PUFs have few CRPs and are typically utilized for on-the-fly secret key generation. Strong PUFs have many CRPs, in the ideal case exponentially increasing with the required chip area, and offer more applications. It should be infeasible to capture all their CRPs in a reasonable time span.

The most prominent strong PUF application is chip authentication whereby only the verifier has to store secret information. In an enrollment phase, the verifier collects arbitrary CRPs from the chip and stores them secretly. In the verification phase, the verifier picks a challenge and requests the PUF response again. The returned response should match the one in the database. A few erroneous bits are typically tolerated hereby as PUF responses are noisy.

The security requirements differ per PUF class. For weak PUFs, it is imperative to keep the responses on chip, as they are post-processed to secret keys by so-called fuzzy extractors [1]. Hardware attacks (invasive, through side channels and via fault injection) should be taken into account. PUFs are often assumed to be resistant against the first category. One can argue that invasion damages the physical structure and hence also the PUF. Experimental evidence is generally lacking however, except for the coating PUF [12]. Electromagnetic radiation is an exploitable side channel for ring oscillator (RO) PUFs [7].

For strong PUFs, CRPs can be obtained by anyone. The security arises from the CRP behavior unpredictability. It should be infeasable to construct a clone via a mathematical model. Modeling through Machine Learning (ML) algorithms, given a training set of CRPs, is a major threat. The arbiter PUF, which quantifies the variability of gate delays, can be modeled as such [6]. Variants of the arbiter PUF which introduce additional non-linearity (XOR, feed-forward, ...) provide more resistance but can still be modeled [8].

Hardware attacks on strong PUFs should be considered too, as they can facilitate modeling. Our main contribution is to show that the noisiness of PUF responses can be exploited as a side channel for modeling. Information leakage through this channel should be considered for all strong PUF designs. We demonstrate a successful attack on 65nm CMOS arbiter PUFs without utilizing any ML algorithm.

This paper is organized as follows. Section II provides the required level of background information to make this text self-sustaining. Meanwhile, the most important aspects of our CMOS implementation are highlighted. Section III describes a model for the repeatability of PUF responses, which is the side channel being exploited. Subsequently, we describe and analyze our attack schemes in section IV. In section V, we present our results for the CMOS implementation and compare them with previous work. Section VI concludes the work.

II. Background

A. Arbiter PUFs

Arbiter PUFs [6] measure structural variability via the propagation delays of logic gates, like ring oscillator [10] and glitch PUFs [11]. The high-level functionality is represented by figure 1. A rising edge propagates through two paths with identically designed delays. Because of nanoscale manufacturing variations however, there is a delay difference Δt_V between both paths. An arbiter decides which path 'wins' the race ($\Delta t_V \lessgtr 0$) and generates a response bit r.

The two paths are constructed from a series of k switching elements. Challenge bits c_i determine for each stage whether path segments are crossed or uncrossed. Each (binary) state of each stage has a unique contribution to Δt_V. So challenge vector $\vec{c}$ determines the arbiter time difference Δt_V and hence the response bit r. The number of CRPs equals 2^k.

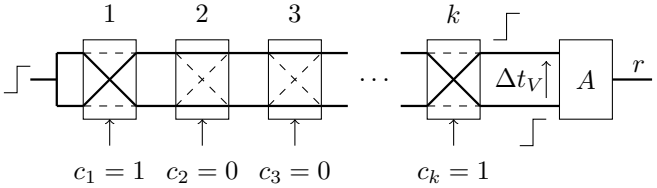


Fig. 1. Arbiter PUF.

CRP-based authentication with strong PUFs requires responses to have multiple bits. A single arbiter PUF produces only a single response bit however. Two solutions, or a mixture of both, are possible. First, one can implement multiple arbiter circuits on the same chip, all having the same challenge as input. Second, one can query a single PUF circuit with multiple challenges and concatenate the responses.

Our 64-stage arbiter PUFs are manufactured in TSMC's 65nm Low Power CMOS technology [3]. A variety of circuits can serve as an arbiter. We chose for a NAND latch, as shown in figure 2. Two cross-coupled NAND gates, implemented in static complementary CMOS logic, determine and store the response bit r. Initially inputs i_1 and i_2 are both zero so that memory nodes r and $\overline{r}$ are both charged. A rising edge will discharge one memory node and simultaneously lock the other.

It is important to match the delay of both NAND gates. Otherwise, bias is introduced and the response bit generation degrades to $\Delta t_V \lessgtr \Delta t_B$. Because the Probability Density Function (PDF) of Δt_V is symmetrical with mean zero, the probability of r to be 1 (or 0) is not 50% anymore. Our 65nm CMOS arbiters are slightly biased in fact because response readout logic is connected to node $\overline{r}$ only, so that is has a higher capacitive load than node r.

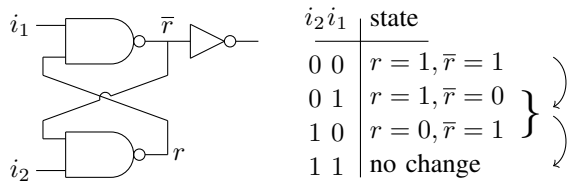


Fig. 2. Arbiter circuit: NAND latch.

B. Modeling Attacks Using Machine Learning

Arbiter PUFs show additive linear behavior which makes them vulnerable to modeling attacks. A single stage can be described by two parameters, one for each challenge bit state, as illustrated in figure 3. The delay difference at the input of stage i flips in sign for the crossed configuration and is incremented with δt_i^1 or δt_i^0 for crossed and uncrossed configurations respectively.

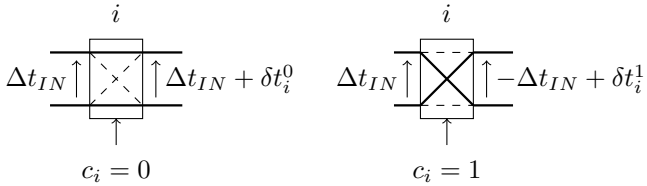


Fig. 3. Modeling.

The impact of a δt on Δt_V is incremental or decremental for an even and odd number of subsequent crossed stages respectively. By lumping together the δt's of neighboring stages, one can model the whole arbiter PUF with only $k+1$ independent parameters (and not $2k$). A formal expression for Δt_V is as follows:

$$\Delta t_V = \vec{\gamma}\vec{\tau} = (\gamma_1 \quad \gamma_2 \quad \dots \quad \gamma_k \quad 1)(\tau_1 \quad \tau_2 \quad \dots \quad \tau_{k+1})^T$$

$$\text{with } \vec{\tau} = \frac{1}{2}\begin{pmatrix} \delta t_1^0 + \delta t_1^1 & + & \delta t_1^0 - \delta t_1^1 \\ & & \delta t_2^0 - \delta t_2^1 \\ & \vdots & \\ \delta t_{k-1}^0 + \delta t_{k-1}^1 & + & \delta t_k^0 - \delta t_k^1 \\ \delta t_k^0 + \delta t_k^1 & & \end{pmatrix} \text{ and}$$

$$\vec{\gamma} = \begin{pmatrix} (1-2c_1)(1-2c_2)\dots(1-2c_{k-1})(1-2c_k) \\ (1-2c_2)\dots(1-2c_{k-1})(1-2c_k) \\ \vdots \\ (1-2c_{k-1})(1-2c_k) \\ (1-2c_k) \\ 1 \end{pmatrix}^T .$$

Vector $\vec{\gamma} \in \{\pm 1\}^{1\times(k+1)}$ is a transformation of challenge vector $\vec{c} \in \{0,1\}^{1\times k}$. Vector $\vec{\tau} \in \mathbb{R}^{(k+1)\times 1}$ contains the lumped stage delays. Arbiter bias can be incorporated too, so that the response bit is still the outcome of $\Delta t_V \lessgtr 0$:

$$\tau_{k+1} = \delta t_k^0 + \delta t_k^1 - \Delta t_B.$$

High modeling accuracies can be obtained through ML techniques like support vector machines and artificial neural networks. Given a limited set of training CRPs, algorithms automatically learn the input-output behavior by trying to generalize the underlying interactions. The more linear a system, the easier to learn its behavior. By using $\vec{\gamma}$ instead of $\vec{c}$ as ML input, a great deal of non-linearity is avoided. The non-linear threshold operation $\Delta t_V \lessgtr 0$ remains however.

In the paper proposing arbiter PUFs as a security primitive, ML was already identified as a threat [6]. They reported a modeling accuracy of 97% for their 64-stage $0.18\mu m$ CMOS implementation. Idem for a more recent $65nm$ implementation, also having 64-bit challenges [2]. Our attacks are performed on the same $65nm$ chip. We circumvent the $\Delta t_V \lessgtr 0$ binarization by exploiting response repeatability as a side channel. A full linear system is obtained, which is straightforward to model.

C. Variability and Noise

The distinction between variability and noise is essential. Both cause deviations with respect to the nominal behavior. Measurements of structural variability, originating from manufacturing processes, are reproducible. One can state that they are defined by spatial distributions (and orientations) of individual molecules of the solid materials. Noise however is a non-reproducible temporal phenomenon. Generally speaking, in electronic circuits, both variability and noise are undesired. PUF circuits measure variability, but are bothered by noise as well, as it reduces the repeatability. In this paper we exploit the presence of noise to characterize the variability relevant for response bit generation.

Both variability and noise are technology dependent. They remain major design and manufacturing challenges, especially while downscaling dimensions according to Moore's law. Random Dopant Fluctuation and Line-Edge/Width Roughness are important sources of variability for CMOS devices [5]. White thermal noise and $1/f$ noise affect the CMOS channel current [4]. Interconnect is affected by Line-Edge/Width Roughness and white thermal noise too.

Environmental conditions like temperature and supply voltage should be kept stable during PUF evaluation. That's because they have an impact on the variability-harvesting PUF behavior.

III. PUF Repeatability Model

Repeatability refers to the short-term reliability of the PUF as affected by CMOS (and interconnect) noise sources. Long-term device aging effects are not included. With $R \in [0, 1]$, we denote the fraction of the responses which evaluates to '1' for a certain CRP. The further from $R = \frac{1}{2}$, the more repeatable.

A. Model Description

The nominal delay difference Δt_V at the arbiter, with respect to the set of all challenges, is assumed to be normally distributed. This distribution has zero mean and standard deviation σ_V, as shown in figure 4a. We collect the device noise of all stages and of the arbiter in one equivalent noise source at the arbiter input. An additional time difference Δt_N is introduced, which we assume to be normally distributed with zero mean and and standard deviation σ_N, as in figure 4b. At the arbiter, we evaluate $\Delta t = (\Delta t_V + \Delta t_N) \lessgtr \Delta t_B$. Fraction R as shown in figure 4c is computed by integrating the noise PDF:

$$R(\Delta t_V) = \frac{1}{2} erfc\left(\frac{\Delta t_B - \Delta t_V}{\sqrt{2}\sigma_N}\right)$$
$$\text{with } erfc(t) = \frac{2}{\sqrt{\pi}} \int_t^\infty e^{-z^2}\, \mathrm{d}z.$$

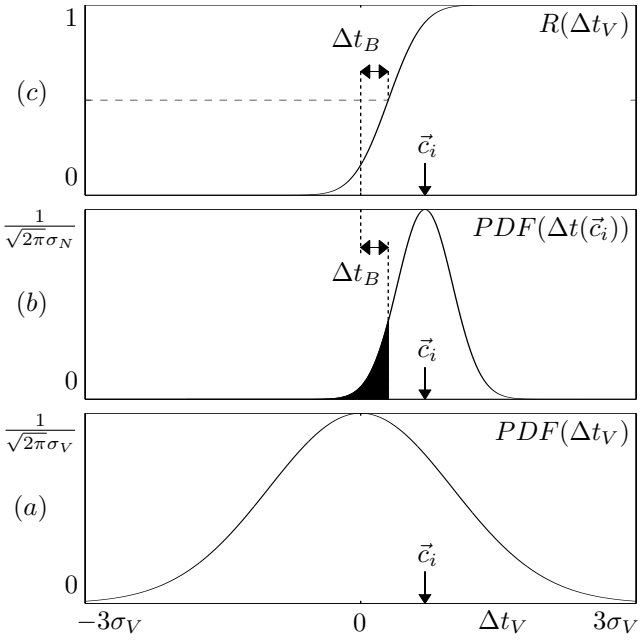


Fig. 4. PUF repeatability model.

The key insight is that repeatability measurements provide direct timing information, as expressed below. Knowledge of neither σ_V nor σ_N is required for modeling purposes. Acquired timing information is all relative, which is not a problem as in the end we are only interested in the sign of $\Delta t_V - \Delta t_B$.

$$\Delta t_V(R) = \Delta t_B - \sqrt{2}\sigma_N erfc^{-1}(2R).$$

B. Repeatability Measurements

Fraction R can be estimated by applying the same challenge many times (parameter M). A measurement error ϵ_R produces an error $\epsilon_{\Delta t_V}$ on the actual arbiter time difference. For small ϵ's, the derivative $\frac{d\Delta t_V}{dR}$ serves as a scaling factor:

$$\epsilon_{\Delta t_V} = \sqrt{2\pi}\sigma_N exp\left(\left(erfc^{-1}(2R)\right)^2\right)\epsilon_R.$$

We distinguish two error phenomena. First, there is the discretization $R \in \{0, \frac{1}{M}, \ldots, \frac{M-1}{M}, 1\}$. The larger M, the less significant this type of errors. Second, there are stochastic errors. We consider a single PUF evaluation as a Bernoulli trial; multiple evaluations then describe a binomial distribution. For simplicity, we could define stochastic error ϵ_R as the standard deviation of the random variable R:

$$\epsilon_R = \sqrt{\frac{R(1-R)}{M}}.$$

Stochastic measurement error ϵ_R has a maximum at $R = \frac{1}{2}$ and decreases monotonically towards $R = 0$ and $R = 1$. Scaling factor $\frac{d\Delta t_V}{dR}$ has an opposing effect and is the most dominant. It has a minimum at $R = \frac{1}{2}$. Towards $R = 0$ and $R = 1$, it increases monotonically and approaches ∞ asymptotically. We prefer measurements around $R = \frac{1}{2}$, but consider the whole $10 - 90\%$ region as reasonable.

C. Model Validation

We validate our model via the PDF of fraction R. An analytical expression, which we match with experimental data, is given below. We measured the reliability of one PUF circuit for 65000 random challenges with $M = 2000$. A normalized histogram serves as PDF. A nonlinear curve fitter iterating over two variables, σ_N/σ_V and $\Delta t_B/\sigma_V$, provides the match. Figure 5 shows an overlay of both PDFs. Only data in the $10 - 90\%$ region has been used for better visibility. Also note the minor bias towards $R = 0$.

$$PDF(R) = PDF(\Delta t_V(R)) \left| \frac{d\Delta t_V}{dR} \right| = \frac{\sigma_N}{\sigma_V} exp \left((erfc^{-1}(2R))^2 - \frac{1}{2} \left(\frac{\Delta t_B - \sqrt{2}\sigma_N erfc^{-1}(2R)}{\sigma_V} \right)^2 \right).$$

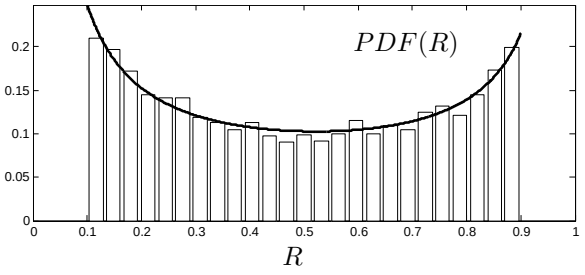


Fig. 5. Model validation.

IV. Modeling Attacks Exploiting Device Noise

A. Least Mean Square (LMS) Method

Figure 4c shows that $R(\Delta t_V)$ is fairly linear for $10\% \leq R \leq 90\%$: we call this the linear region. As Δt_V is a linear combination of model parameters τ_1 to τ_{k+1}, so is R in the linear region (approximately). Consider a set of N training CRPs in that region where each response is evaluated M times: $\{\vec{c}_i,\ R_i\}$. For $N \geq k + 1$, we can simply solve the (overdetermined) system of linear equations shown below in a Least Mean Square manner. Numerically stable algorithms are described in literature.

$$\mathbf{\Gamma}\vec{\tau} = \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_N \end{pmatrix} \quad \text{with } \mathbf{\Gamma}^{N\times(k+1)} = \begin{pmatrix} \vec{\gamma}_1 \\ \vec{\gamma}_2 \\ \vdots \\ \vec{\gamma}_N \end{pmatrix}.$$

As suggested earlier, arbiter bias is included in element τ_{k+1}. To predict the PUF response for a challenge $\vec{c}$, one should check whether $R = \vec{\gamma}\vec{\tau} \lessgtr \frac{1}{2}$. The predicted value of R can also be utilized to estimate the prediction certainty: the further from $\frac{1}{2}$, the better. This feature is not intrinsically available for ML techniques like artificial neural networks.

One could improve the linearity by applying the transformation below. For response prediction, one should check whether $\vec{\gamma}\vec{\tau} \lessgtr erfc^{-1}(1) = 0$. We do not apply this transformation as we observe only very minor improvements for the modeling accuracy.

$$\mathbf{\Gamma}\vec{\tau} = \begin{pmatrix} -erfc^{-1}(2R_1) \\ -erfc^{-1}(2R_2) \\ \vdots \\ -erfc^{-1}(2R_N) \end{pmatrix}.$$

A major speed bottleneck is that most CRPs are very repeatable and hence not suitable for modeling purposes. We estimated that only 10.2% of the CRPs belong to the linear region (averaged over 32 PUF instances, 20000 CRPs each). Note that figure 5 already provided a rough estimate of this fraction.

B. Differential Measurements Method

We present a second attack scheme which estimates the elements of model vector $\vec{\tau}$ one by one. Its performance and usability are in all aspects inferior to the LMS method. High modeling accuracies ($> 95\%$) can be obtained too, but much more PUF evaluations are therefore required. A description of this method is very useful though. Additional insights are provided and the essential idea might be recyclable for attacking other strong PUFs. Results are not discussed. Consider the following arbitrary challenge as a reference:

$$\begin{cases} \vec{c}_{REF} = (c_1 & c_2 & \dots & c_k) \\ \vec{\gamma}_{REF} = (\gamma_1 & \gamma_2 & \dots & \gamma_k & 1). \end{cases}$$

One can choose challenges $\vec{c}_i$ so that $\vec{\gamma}_{REF}$ and $\vec{\gamma}_i$ only differ in element i, with $i \in [1\ k]$:

$$\begin{cases} \vec{c}_1 = (\overline{c}_1 & c_2 & \dots & c_k) \\ \vec{\gamma}_1 = (-\gamma_1 & \gamma_2 & \dots & \gamma_k & 1) \end{cases}$$

$$\begin{cases} \vec{c}_2 = (\overline{c}_1 & \overline{c}_2 & \dots & c_k) \\ \vec{\gamma}_2 = (\gamma_1 & -\gamma_2 & \dots & \gamma_k & 1) \end{cases}$$

$$\vdots$$

$$\begin{cases} \vec{c}_k = (c_1 & \dots & \overline{c}_{k-1} & \overline{c}_k) \\ \vec{\gamma}_k = (\gamma_1 & \dots & \gamma_{k-1} & -\gamma_k & 1). \end{cases}$$

Only τ_i then contributes to the difference of both arbiter time differences:

$$\Delta t_{V,REF} - \Delta t_{V,i} = (\vec{\gamma}_{REF} - \vec{\gamma}_i)\vec{\tau} = 2\gamma_i\tau_i.$$

In the linear region, τ_i is hence proportional with the difference of fraction R. When applying the same arbitrary scaling factor as for the LMS method, one can estimate τ_i as shown below. Reusing repeatability measurements across different element of $\vec{\tau}$, as suggested above, can provide a method speed-up.

$$\tau_i = \frac{\gamma_i}{2}\Delta R_i \quad \text{with } \Delta R_i = R_{REF} - R_i.$$

Similar as for the LMS method, one could apply a correction for the non-linearity:

$$\tau_i \propto \gamma_i \left(erfc^{-1}(2R_i) - erfc^{-1}(2R_{REF}) \right).$$

Estimating τ_{k+1} is more complicated as arbiter bias cancels out for differential measurements. One could estimate τ_1 to τ_k first and subsequently determine τ_{k+1} so that the modeling error with respect to a CRP verification set is minimized.

The usability of the method is somewhat limited for elements of $\vec{\tau}$ with large magnitude. Unfortunately, they are the most important ones for predicting PUF responses. It might be difficult or even impossible to fit two repeatability measurements in the linear region. So in the worst case, only a lower bound of τ_i can be provided. We encountered typically a few problematic τ_i's per arbiter circuit. As a solution, one could estimate a smaller time difference $\tau_i \pm \tau_j$ instead, with τ_j estimated before. Remark that the estimation error of τ_j does propagate hereby.

C. Accuracy Analysis

We analyze and compare the accuracy of both methods. The linear equations of the differential measurements method are rewritten below, in conformity with the LMS method notation. To avoid unnecessary complications, we ignore potential issues with large τ_i magnitudes. We also ignore the fact that τ_{k+1} is computed differently.

$$\mathbf{\Gamma}\vec{\tau} = \frac{1}{2}\begin{pmatrix} \Delta R_1 \\ \Delta R_2 \\ \vdots \\ \Delta R_{k+1} \end{pmatrix} \quad \text{with } \mathbf{\Gamma} = \begin{pmatrix} \pm 1 & 0 & \dots & 0 \\ 0 & \pm 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & \pm 1 \end{pmatrix}.$$

For both methods, errors in the repeatability measurements cause an error on PUF model vector $\vec{\tau}$, as shown below. The magnitude of $\vec{\epsilon}_R$ is typically a factor $\sqrt{2}/2$ smaller for the differential measurements method.

$$\mathbf{\Gamma}(\vec{\tau} + \vec{\epsilon}_\tau) = (\vec{R} + \vec{\epsilon}_R) \quad \text{with } \mathbf{\Gamma}\vec{\tau} = \vec{R}.$$

To quantify the extent to which repeatability errors propagate, one could compute the condition number κ_C of $\mathbf{\Gamma}$. In our case, a modified definition is much more appropriate. Model error $\vec{\epsilon}_\tau = \mathbf{\Gamma}^+\vec{\epsilon}_R$ is determined by the (pseudo)inverse of challenge matrix $\mathbf{\Gamma}$. Row i of $\mathbf{\Gamma}^+$ determines for τ_i to which extent repeatability errors accumulate/cancel out. We sum the elements in each row and average their absolute values:

$$\kappa_C = \frac{1}{k+1}\sum_{i=1}^{k+1}\left|\sum_{j=1}^{N}\mathbf{\Gamma}^+_{i,j}\right|.$$

For the differential measurements method, the condition number is always one. We performed a simulation for the LMS method, as shown in figure 6. For different values of $N \geq k+1$, we generated 1000 random challenge matrices and averaged their condition numbers. Condition number κ_C decreases monotonically with increasing N: rapidly in the beginning and increasingly slower afterwards. As demonstrated later in section V, very small values of M become feasible as quantization errors are canceling out efficiently. The overall performance is clearly superior to the differential measurements method.

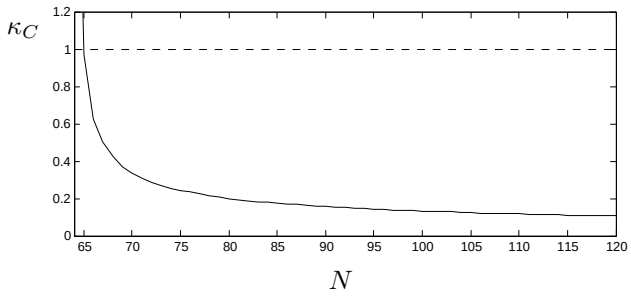


Fig. 6. Condition number of the LMS method.

D. Query Algorithms

Only 10.2% of the CRPs are usable for modeling purposes, which is the main speed bottleneck for our attacks. One might try to increase this fraction by an adaptive query algorithm. As discussed for the differential measurements method, one can perform small steps of Δt_V. So a CRP in the linear region might still be there after making such a small step. For optimal performance, one should learn while querying.

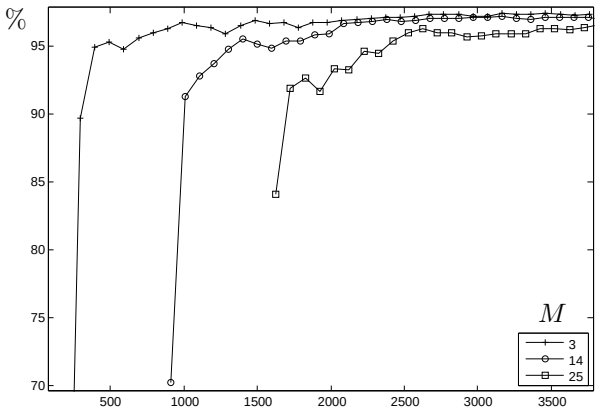
We do not implement such an algorithm because it has some major drawbacks. First, one should take into account the method by which multiple response bits are generated. Modeling a single PUF is more advantageous than modeling many parallel PUFs. That's because query algorithms can adapt their behavior to maximally benefit only a single PUF. Second, in case of the LMS method, the rows of $\mathbf{\Gamma}$ become strongly correlated, which increases its condition number. Furthermore, our paper only provides a proof-of-concept for the repeatability side channel and is rather an ally than a competitor for provenly fast ML approaches.

V. Results and Discussion

We want an accurate model while keeping the number of PUF evaluations low. Three factors contribute to the total number of PUF evaluations: the fraction of CRPs belonging to the linear region, the number of measured CRPs in that region (N) and the number of PUF evaluations per CRP (M). We measure modeling accuracy via a verification set of 5000 randomly chosen CRPs. Each verification challenge was evaluated $M = 100$ times on 32 PUF instances, with $\frac{M}{2} = 50$ as a response bit decision threshold.

Figure 7 demonstrates that the best trade-off is obtained for very low values of M. That's because quantization errors are dealt with efficiently, as mentioned earlier. We plot the modeling accuracy versus the number of PUF evaluations in the linear region (NM) for $M \in \{3, 14, 25\}$. The first model can be constructed for $N = k+1 = 65$ and the modeling accuracy ramps up rapidly with increasing N afterwards. This behavior is conform with the condition number simulation in figure 6.

Table I provides numerical values of the modeling accuracy for $M \in \{3, 5, 7\}$. The accuracy averaged over 32 PUF instances and its accompanying standard deviation are listed. Note that low values of M might enable an eavesdropping attack in case of a majority vote, depending on whether the vote is performed before or after response transmission.



Number of PUF evaluations in the linear region (NM)

Fig. 7. Modeling accuracy of the LMS method.

	M		
N	3	5	7
100	$88.08\% \pm 5.04\%$	$92.36\% \pm 2.61\%$	$93.77\% \pm 1.52\%$
200	$94.20\% \pm 2.27\%$	$95.87\% \pm 1.07\%$	$96.40\% \pm 0.83\%$
300	$95.26\% \pm 1.28\%$	$96.57\% \pm 0.82\%$	$96.96\% \pm 0.66\%$
400	$95.81\% \pm 0.87\%$	$96.93\% \pm 0.63\%$	$97.22\% \pm 0.57\%$
500	$96.14\% \pm 0.90\%$	$97.11\% \pm 0.67\%$	$97.37\% \pm 0.58\%$
600	$96.39\% \pm 0.72\%$	$97.29\% \pm 0.57\%$	$97.44\% \pm 0.56\%$
700	$96.58\% \pm 0.68\%$	$97.40\% \pm 0.57\%$	$97.54\% \pm 0.53\%$
800	$96.73\% \pm 0.59\%$	$97.43\% \pm 0.58\%$	$97.56\% \pm 0.54\%$

TABLE I. MODELING ACCURACY OF THE LMS METHOD.

The fraction of CRPs belonging to the linear region is approximately constant, except for low values of M. Because of the discretization of R, there is an asymmetry of the available bins around $R = 10\%$ as well as $R = 90\%$. For $M = 3$, 5 and 7, the fraction of usable CRPs is 6.6%, 9.0% and 10.4% respectively (averaged over 32 PUF instances, 20000 CRPs each). For large values of M, this fraction is about 10.2%.

From previous paragraph and table I, we conclude that an over 95%-accurate model can be constructed with less than 15000 PUF evaluations. ML techniques require less than 5000 PUF evaluations and are hence faster, as confirmed by measurements on the same chip [2]. The large amount of repeatable responses is our main speed bottleneck: when taking only usable CRPs into account, our method would be faster. Furthermore, an eavesdropping attack is always possible with ML, so it is still the recommended approach to model arbiter PUFs.

We propose joined efforts instead of competition however. The response repeatability side channel could facilitate a ML attack. Analog information is obtained from digital response bits and could be used as ML input. The arbiter PUF, its variants (XOR, feed-forward, ...) and other strong PUF designs might be more vulnerable than with ML only. We propose this analysis as further work.

VI. CONCLUSION

Response repeatability can be exploited as a side channel for modeling strong PUFs. Information leakage through this side channel should be considered for all strong PUF designs. As a proof-of-concept, we were able to successfully model 65nm arbiter PUFs without utilizing any ML algorithm. We propose a combined attack strategy in which the device noise side channel could facilitate ML.

ACKNOWLEDGMENT

This work was supported in part by the European Commission through the ICT programme under contract FP7-ICT-2011-317930 HINT. In addition this work is supported by the Research Council of KU Leuven: GOA TENSE (GOA/11/007), by the Flemish iMinds projects, by the Flemish Government through FWO G.0550.12N and the Hercules Foundation AKUL/11/19. Jeroen Delvaux is funded by IWT-Flanders grant no. 121552.

REFERENCES

[1] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97-139, Mar. 2008.

[2] G. Hospodar, R. Maes, and I. Verbauwhede, "Machine Learning Attacks on 65nm Arbiter PUFs: Accurate Modeling poses strict Bounds on Usability," in *IEEE International Workshop on Information Forensics and Security*, WIFS 2012, pp. 37-42, Dec. 2012.

[3] P. Koeberl, R. Maes, V. Rožić, V. Van der Leest, E. Van der Sluis, and I. Verbauwhede, "Experimental Evaluation of Physically Unclonable Functions in 65 nm CMOS," in *IEEE European Solid-State Circuits Conference*, ESSCIRC 2012, pp. 486-489, Sep. 2012.

[4] A. Konczakowska and B. M. Wilamowski, "Noise in Semiconductor Devices," Industrial Electronics Handbook, vol. 1 Fundamentals of Industrial Electronics, 2nd Edition, chapter 11, CRC Press 2011.

[5] K. Kuhn, C. Kenyon, A. Kornfeld, M. Liu, A. Maheshwari, W. Shih, S. Sivakumar, G. Taylor, P. VanDerVoorn and K. Zawadzki, "Managing Process Variation in Intel's 45nm CMOS Technology," *Intel Technology Journal*, vol. 12, no. 2, pp. 92-110, Jun. 2008.

[6] J.W. Lee, D. Lim, B. Gassend, G.E. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *IEEE Symposium on VLSI Circuits*, VLSIC 2004, pp. 176-179, Jun. 2004.

[7] D. Merli, D. Schuster, F. Stumpf and G. Sigl, "Semi-invasive EM attack on FPGA RO PUFs and countermeasures," in *Workshop on Embedded Systems Security*, WESS 2011, pp 1-9, Oct. 2011.

[8] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *ACM conference on Computer and Communications Security*, CCS 2010, pp. 237-249, Oct. 2010.

[9] U. Rührmair, S. Devadas and F. Koushanfar, "Security based on Physical Unclonability and Disorder," Introduction to Hardware Security and Trust, Springer, Book Chapter, 2011.

[10] G.E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *IEEE Design Automation Conference*, DAC 2007, pp. 9-14, Jun. 2007.

[11] D. Suzuki and K. Shimizu, "The Glitch PUF: A New Delay-PUF Architecture Exploiting Glitch Shapes," in *Cryptographic Hardware and Embedded Systems*, CHES 2010, pp. 366-382, Aug. 2010.

[12] P. Tuyls, G.J. Schrijen, B. Skoric, J.V. Geloven, N. Verhaegh and R. Wolters, "Read-Proof Hardware from Protective Coatings," in *Cryptographic Hardware and Embedded Systems*, CHES 2006, pp. 369-383, Oct. 2006.